A Bridge Between the Asynchronous Message Passing Model and Local Computations in Graphs

Jérémie Chalopin, Yves Métivier

A Bridge between the Asynchronous Message Passing Model and Local Computations in Graphs (Extended Abstract)

Jérémie Chalopin and Yves Métivier

{chalopin,metivier}@labri.fr
LaBRI Université Bordeaux 1, ENSEIRB,
351 cours de la Libération
33405 Talence, France

1 Introduction

A distributed system is a collection of processes that can interact. Three major process interaction models in distributed systems have principally been considered: - the message passing model, - the shared memory model, - the local computation model. In each model the processes are represented by vertices of a graph and the interactions are represented by edges. In the message passing model and the shared memory model, processes interact by communication primitives: messages can be sent along edges or atomic read/write operations can be performed on registers associated with edges. In the local computation model interactions are defined by labelled graph rewriting rules; supports of rules are edges or stars. These models (and their sub-models) reflect different system architectures, different levels of synchronization and different levels of abstraction. Understanding the power of various models, the role of structural network properties and the role of the initial knowledge enhances our understanding of basic distributed algorithms. This is done with some typical problems in distributed computing: election, naming, spanning tree construction, termination detection, network topology recognition, consensus, mutual exclusion. Furthermore, solutions to these problems constitute primitive building blocks for many other distributed algorithms. A survey may be found in [FR03], this survey presents some links with several parameters of the models including synchrony, communication media and randomization. An important goal in the study of these models is to understand some relationships between them. This paper is a contribution to this goal; more precisely we establish a bridge between tools and results presented in [YK96] for the message passing model and tools and results presented in [Ang80,BCG$^+$96,Maz97,CM04,CMZ04,Cha05] for the local computation model.

In the message passing model studied by Yamashita and Kameda in [YK96], basic events are: send events, receive events, internal events and transmission events. They have obtained characterizations of graphs permitting a leader election algorithm, a spanning tree construction algorithm and a topology recognition algorithm. For this, they introduced the concept of view. The view from a

vertex v of a graph G is an infinite labelled tree rooted in v obtained by considering all labelled walks in G starting from v. The characterizations use also the notion of symmetricity. The symmetricity of a graph depends on the number of vertices that have the same view. The local computation model has been studied intensively since the pioneer work of Angluin [Ang80]. A basic event changes the state attached to one vertex or the states of a group of neighbouring vertices. The new state depends on the state of one neighbour or depends on the states of a group of neighbours (some examples are presented in [RFH72,BV99,BCG$^+$96]). Characterizations of graphs, for the existence of an election algorithm, have been obtained using classical combinatorial material like the notions of fibration and of covering: special morphisms which ensure isomorphism of neighbourhoods of vertices or arcs. Some effective characterizations of computability of relations in anonymous networks using fibrations and views are given in [BV01]. The new state of a vertex must depend on the previous state and on the states of the in-neighbours.

The Election Problem and the Naming Problem. The election problem is one of the paradigms of the theory of distributed computing. It was first posed by LeLann [LeL77]. A distributed algorithm solves the election problem if it always terminates and in the final configuration exactly one process is marked as *elected* and all the other processes are *non-elected*. Moreover, it is supposed that once a process becomes *elected* or *non-elected* then it remains in such a state until the end of the algorithm. Election algorithms constitute a building block of many other distributed algorithms. The naming problem is another important problem in the theory of distributed computing. The aim of a naming algorithm is to arrive at a final configuration where all processes have unique identities. Being able to give dynamically and in a distributed way unique identities to all processes is very important since many distributed algorithms work correctly only under the assumption that all processes can be unambiguously identified. The enumeration problem is a variant of the naming problem. The aim of a distributed enumeration algorithm is to assign to each network vertex a unique integer in such a way that this yields a bijection between the set $V(G)$ of vertices and $\{1, 2, \ldots, |V(G)|\}$.

The Main Results. In Section 3 we introduce a new labelled directed graph which encodes a network in which processes communicate by asynchronous message passing with a symmetric port numbering. The basic events (send, receive, internal, transmission) are encoded by local computations on arcs. From this directed graph, we deduce necessary conditions for the existence of an election (and a naming) algorithm on a network (Proposition 4). The conditions are also sufficient (Theorem 1): we give a naming (and an election) algorithm in Section 5 (Algorithm 1). This algorithm is totally asynchronous (the Yamashita and Kameda algorithm needs a pseudo-synchronization). Furthermore, our algorithm does not need the FIFO property of channels (i.e., it does not require that messages are received in the same order as they have been sent). The size of the buffer does not interfere in the impossibility proof for the election. Moreover, we present a fully polynomial algorithm. Given a graph G with n vertices and m

edges, in Yamashita and Kameda algorithm the size of each message can be 2^n whereas in our algorithm the size is bounded by $O(m \log n)$ and the number of messages is $O(m^2 n)$. Another consequence of this bridge between these models is a direct characterization of graphs having a symmetricity equal to 1 in the sense of Yamashita and Kameda using the notion of covering. The same techniques can be applied to some other problems such as spanning tree computation or the topology recognition problem. We can note also that our algorithm may elect even if the necessary condition is not verified: in this case an interesting problem is the study of the probability of this event.

2 Preliminaries

The notations used here are essentially standard. The definitions and main properties are presented in [BV02]. We consider finite, undirected, connected graphs having possibly self-loops and multiple edges, $G = (V(G), E(G), \text{Ends})$, where $V(G)$ denotes the set of vertices, $E(G)$ denotes the set of edges and Ends is a map assigning to every edge two vertices: its ends. A symmetric digraph (V, A, s, t) is a digraph endowed with a symmetry, that is, an involution $Sym : A \to A$ such that for every $a \in A : s(a) = t(Sym(a))$. Labelled graphs will be designated by bold letters like $\mathbf{G}$, $\mathbf{H}$, ... If $\mathbf{G} = (G, \lambda)$ is a labelled graph then G denotes the underlying graph and λ denotes the labelling function. The labelling may encode any initial process knowledge. Examples of such knowledge include: (a bound on) the number of processes, (a bound on) the diameter of the graph, the topology, identities or partial identities, distinguished vertices. The notion of fibration and of of covering are fundamental in this work.

Definition 1. *A fibration between the digraphs D and D' is a morphism φ from D to D' such that for each arc a' of $A(D')$ and for each vertex v of $V(D)$ such that $\varphi(v) = v' = t(a')$ there exists a unique arc a in $A(D)$ such that $t(a) = v$ and $\varphi(a) = a'$.*

The arc a is called the lifting of a' at v, D is called the total digraph and D' the base of φ. We shall also say that D is fibred (over D'). In the sequel directed graphs are always strongly connected and total digraphs non empty thus fibrations will be always surjective.

Definition 2. *An opfibration between the digraphs D and D' is a morphism φ from D to D' such that for each arc a' of $A(D')$ and for each vertex v of $V(D)$ such that $\varphi(v) = v' = s(a')$ there exists a unique arc a in $A(D)$ such that $s(a) = v$ and $\varphi(a) = a'$. A covering projection is a fibration that is also an opfibration.*

If a covering projection $\varphi : D \to D'$ exists, D is said to be a covering of D' via φ. Covering projections verify:

Proposition 1. *A covering projection $\varphi : D \to D'$ with a connected base and a nonempty covering is surjective; moreover, all the fibres have the same cardinality. This cardinality is called the number of sheets of the covering.*

A digraph D is covering prime if there is no digraph D' not isomorphic to D such that D is a covering of D' (i.e., D is a covering of D' implies that D is isomorphic to D'). Let D and D' be two digraphs such that D is a surjective covering of D' via φ. If D' has no self-loop then for each arc $a \in A(D) : \varphi(s(a)) \neq \varphi(t(a))$. Finally the following property is a direct consequence of the definitions and it is fundamental in the sequel of this paper :

Proposition 2. *Let D and D' be two digraphs such that D' has no self-loop and D is a surjective covering of D' via φ. If $a_1 \neq a_2$ and $a_1, a_2 \in \varphi^{-1}(a')$ $(a' \in A(D'))$ then $Ends(a_1) \cap Ends(a_2) = \emptyset$.*

The notions of fibrations and of coverings extend to labelled digraphs in an obvious way: the morphisms must preserve the labelling. Examples of coverings are given in Figures 1 and 2.

Local Computations on Arcs. In this paper we consider labelled digraphs and we assume that local computations modify only labels of vertices. Digraph relabelling systems on arcs and more generally local computations on arcs satisfy the following constraints, that arise naturally when describing distributed computations with decentralized control: -($C1$) they do not change the underlying digraph but only the labelling of vertices, the final labelling being the result of the computation (*relabelling relations*), -($C2$) they are *local*, that is, each relabelling step changes only the label of the source and the label of the target of an arc, -($C3$) they are *locally generated*, that is, the applicability of a relabelling rule on an arc only depends on the label of the arc, the labels of the source and of the target (locally generated relabelling relation). The relabelling is performed until no more transformation is possible, i.e., until a normal form is obtained. Let $\mathcal{R}$ be a locally generated relabelling relation, $\mathcal{R}^*$ stands for the reflexive-transitive closure of $\mathcal{R}$. The labelled digraph $\mathbf{D}$ is $\mathcal{R}$-*irreducible* (or just irreducible if $\mathcal{R}$ is fixed) if there is no $\mathbf{D_1}$ such that $\mathbf{D}\ \mathcal{R}\ \mathbf{D_1}$.

3 From Asynchronous Message Passing to Local Computations on Arcs

The model. Our model follows standard models for distributed systems given in [AW98,Tel00]. The communication model is a point-to-point communication network which is represented as a simple connected undirected graph where vertices represent processes and two vertices are linked by an edge if the corresponding processes have a direct communication link. Processes communicate by message passing, and each process knows from which channel it receives a message or it sends a message. An edge between two vertices v_1 and v_2 represents a channel connecting a port i of v_1 to a port j of v_2. We consider the asynchronous message passing model: processes cannot access a global clock and a message sent from a process to a neighbour arrives within some finite but unpredictable time.

From Undirected Labelled Graphs to Labelled Digraphs. A first approximation of a network, with knowledge about the structure of the underlying

graph, is a simple labelled graph $\mathbf{G} = (V(\mathbf{G}), E(\mathbf{G}))$. We associate to this undirected labelled graph a labelled digraph $\overleftrightarrow{\mathbf{G}} = (V(\overleftrightarrow{\mathbf{G}}), A(\overleftrightarrow{\mathbf{G}}))$ defined in the following way. (This construction is illustrated in Figure 1 and in Figure 2).

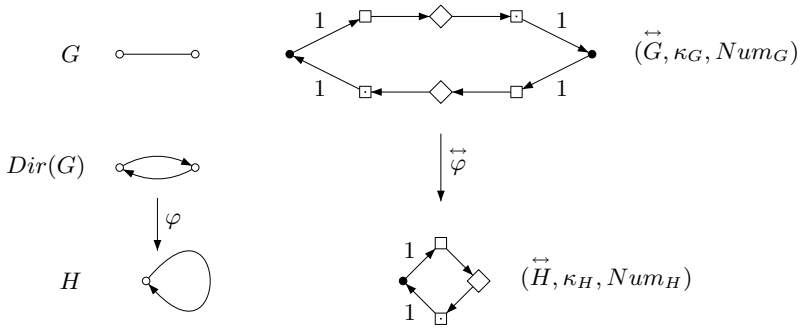

Fig. 1. We adopt the following notation conventions for vertices of $(\overleftrightarrow{G}, \kappa_G, Num_G)$ and $(\overleftrightarrow{H}, \kappa_H, Num_H)$. A black-circle vertex corresponds to the label **process**, a square vertex corresponds to the label **send**, a diamond vertex corresponds to the label **transmission**, and a square-dot vertex corresponds to the label **receive**. The digraph $(\overleftrightarrow{G}, \kappa_G, Num_G)$ is a covering of $(\overleftrightarrow{H}, \kappa_H, Num_H)$ and the port numbering is symmetric. Thus there is no election algorithm for G.

Let u and v be two vertices of $\mathbf{G}$ such that u and v are neighbours, we associate to the edge $\{u, v\}$ the set $V_{\{u,v\}}$ of 6 vertices denoted $\{outbuf(u,v), t(u,v), inbuf(u,v), outbuf(v,u), t(v,u), inbuf(v,u)\}$, and the set $A_{\{u,v\}}$ of 8 arcs defined by: $\{(u, outbuf(u,v)), (outbuf(u,v), t(u,v)), (t(u,v), inbuf(u,v)), (inbuf(u,v), v), (v, outbuf(v,u)), (outbuf(v,u), t(v,u)), (t(v,u), inbuf(v,u)), (inbuf(v,u), u)\}$.

Finally, $V(\overleftrightarrow{\mathbf{G}}) = V(G) \cup (\bigcup_{\{u,v\} \in E(G)} V_{\{u,v\}})$ and $A(\overleftrightarrow{\mathbf{G}}) = \bigcup_{\{u,v\} \in E(G)} A_{\{u,v\}}$.

The arc $(u, outbuf(u,v))$ is denoted $out(u,v)$, $receiver(out(u,v))$ is the vertex v, and the arc $(inbuf(v,u), u)$ is denoted by $in(v,u)$.

If $\mathbf{G} = (G, \lambda)$ then $\overleftrightarrow{\mathbf{G}} = (\overleftrightarrow{G}, \lambda_{\overleftrightarrow{G}})$ where $\lambda_{\overleftrightarrow{G}}(v) = \lambda(v)$ for each $v \in V(G)$.

In the sequel we consider digraphs obtained by this construction; in general networks are anonymous: vertices have no name. Nevertheless we need to memorize the meaning (semantic) of vertices thus we label vertices of $\overleftrightarrow{\mathbf{G}}$ with a labelling function κ, the set of labels is: $\{$**process**, **send**, **receive**, **transmission**$\}$, - if a vertex x of $V(\overleftrightarrow{\mathbf{G}})$ corresponds to a vertex u of $V(G)$ then $\kappa(x) =$ **process**, - if a vertex x of $V(\overleftrightarrow{\mathbf{G}})$ corresponds to a vertex of the form $outbuf(u,v)$ then $\kappa(x) =$ **send**, - if a vertex x of $V(\overleftrightarrow{\mathbf{G}})$ corresponds to a vertex of the form $inbuf(u,v)$ then $\kappa(x) =$ **receive**, - if a vertex x of $V(\overleftrightarrow{\mathbf{G}})$ corresponds to a vertex of the form $t(u,v)$ then $\kappa(x) =$ **transmission**. Using a new label *neutral*, κ is extended to $(V(\overleftrightarrow{\mathbf{G}}), A(\overleftrightarrow{\mathbf{G}}))$. We denote by $\mathcal{E}$ the map which associates to a labelled graph $\mathbf{G}$ the labelled digraph $\mathcal{E}(\mathbf{G}) = (\overleftrightarrow{\mathbf{G}}, \kappa)$ described above.

Two adjacent vertices of $\mathcal{E}(\mathbf{G}) = (\overleftrightarrow{\mathbf{G}}, \kappa)$ have different labels thus if the digraph $\mathcal{E}(\mathbf{G}) = (\overleftrightarrow{\mathbf{G}}, \kappa)$ is a covering of a digraph $\mathbf{D}$ then $\mathbf{D}$ has no self-loop.

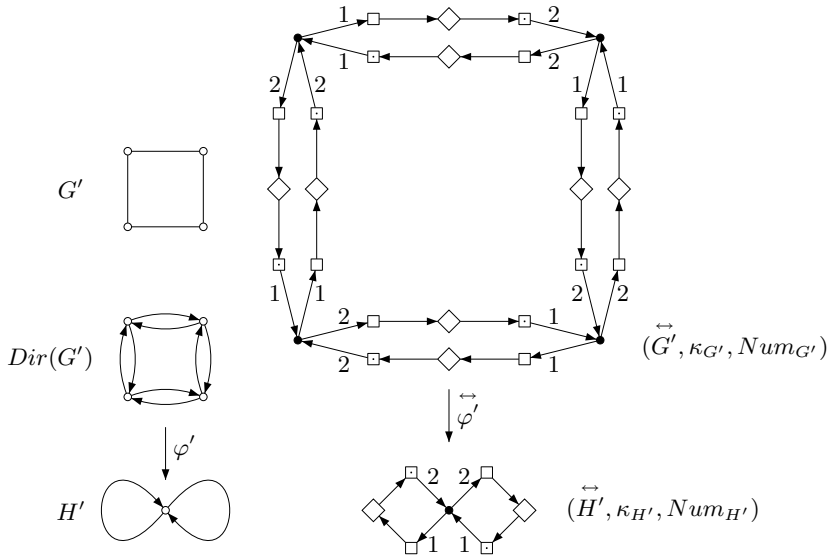

Fig. 2. With the notation conventions of Figure 1, we deduce from the covering relation and the symmetry of the port numbering that there is no election algorithm for the graph G'. With the same argument the same result is obtained for any ring.

Remark 1. By the insertion of special vertices in arcs and the labelling of vertices, we define a transformation $\mathcal{E}'$ such that $(\overleftrightarrow{\mathbf{G}}, \kappa)$ can be obtained directly from $Dir(\mathbf{G})$, i.e., $\mathcal{E}'(Dir(\mathbf{G})) = (\overleftrightarrow{\mathbf{G}}, \kappa)$. Furthermore if $Dir(\mathbf{G})$ is a covering of a labelled digraph $\mathbf{D}$ then $(\overleftrightarrow{\mathbf{G}}, \kappa)$ is a covering of $\mathcal{E}'(\mathbf{D})$.

Port Numbering and Symmetric Port Numbering. We can notice, that for a digraph $\mathcal{E}(\mathbf{G}) = (\overleftrightarrow{\mathbf{G}}, \kappa)$, if we consider a vertex x labelled **process** then $deg^+(x) = deg^-(x)$. Each process knows from which channel it receives a message or it sends a message, that is, each process assigns numbers to its ports. Thus we consider a labelling Num of arcs of $\mathcal{E}(\mathbf{G})$ coming into or going out of vertices labelled **process** such that for each vertex x labelled **process** the restriction of Num assigns to each outgoing arc a unique integer of $[1, deg^+(x)]$ and assigns to each arc coming into a unique integer of $[1, deg^-(x)]$, such a labelling is a local enumeration of arcs incident to **process** vertices and it is called a port numbering. In a message passing system the communication is done over communication channels. A channel provides a bidirectional connection between two processes. Finally, the topology is encoded by an undirected graph G where an edge corresponds to a channel. Let v be a vertex of G, the port numbering

for the vertex v is defined by an enumeration of edges incident to the vertex v, this enumeration induces an enumeration of the arcs of $(\overleftrightarrow{\mathbf{G}}, \kappa)$. This enumeration is symmetric, i.e., Num verifies for each arc of the form $out(u,v)$: $Num(out(u,v)) = Num(in(v,u))$; this condition is called the symmetry of the port numbering (or equivalently of Num). Such a port numbering is said symmetric. Again, using the special label *neutral*, Num is considered as a labelling function of $\mathcal{E}(\mathbf{G})$. The graph $(\overleftrightarrow{\mathbf{G}}, \kappa, Num)$ is denoted by $\mathcal{H}(\mathbf{G})$. The hypothesis of the symmetry of the port numbering is done in [YK96] and it corresponds to the complete port awareness model in [BCG$^+$96].

Basic Instructions. As in [YK96] (see also [Tel00] pp. 45-46), we assume that each process, depending on its state, either changes its state, or receives a message via a port or sends a message via a port. Let $Inst$ be this set of instructions. This model is equivalent to the model of local computations on arcs with respect to the initial labelling as it is depicted in the following remark.

Remark 2. Let $\mathbf{G}$ be a labelled graph, let $\mathcal{H}(\mathbf{G}) = (\overleftrightarrow{\mathbf{G}}, \kappa, Num)$ be the labelled digraph obtained from $\mathbf{G}$. The labelled digraph $\mathcal{H}(\mathbf{G})$ enables to encode the following events using local computations on arcs: - an internal event "a process changes its state" can be encoded by a relabelling rule concerning a vertex labelled **process**, - a send event "the process x sends a message via the port i" can be encoded by a relabelling rule concerning an arc of the form (x,y) with $\kappa(x) =$ **process**, $\kappa(y) =$ **send** and $Num((x,y)) = i$, - a receive event "the process y receives a message via the port i" can be encoded by a relabelling rule concerning an arc of the form (x,y) with $\kappa(x) =$ **receive**, $\kappa(y) =$ **process** and $Num((x,y)) = i$, - an event concerning the transmission control can be encoded by a relabelling rule concerning an arc of the form (x,y) or (y,z) with $\kappa(x) =$ **send**, $\kappa(y) =$ **transmission** and $\kappa(z) =$ **receive**.

The Election and the Naming Problems. Consider a network $\mathbf{G}$ with a symmetric port numbering Num. An algorithm $\mathcal{A}$ is an election algorithm for $(\overleftrightarrow{\mathbf{G}}, \kappa, Num)$ if each execution of $\mathcal{A}$ on $\mathbf{G}$ with the port numbering Num successfully elects a process. We are particularly interested in characterizing the networks that admit an election algorithm whatever the symmetric port numbering is. We say that an algorithm $\mathcal{A}$ is an election algorithm for a graph $\mathbf{G}$ if for each symmetric port numbering Num, $\mathcal{A}$ is an election algorithm for $(\overleftrightarrow{\mathbf{G}}, \kappa, Num)$. We will use the same conventions for the naming problem.

4 A Necessary Condition for the Election Problem and the Naming Problem

First, we present a fundamental lemma which connects coverings and locally generated relabelling relations on arcs. It is the natural extension of the Lifting Lemma [Ang80] and it is a direct consequence of Proposition 2.

Lemma 1 (Lifting Lemma). *Let $\mathcal{R}$ be a locally generated relabelling relation on arcs and let $\mathbf{D}_1$ be a covering of the digraph $\mathbf{D}'_1$ via the morphism γ; we*

assume that $\mathbf{D}'_1$ *has no self-loop. If* $\mathbf{D}'_1 \ \mathcal{R}^* \ \mathbf{D}'_2$ *then there exists* $\mathbf{D}_2$ *such that* $\mathbf{D}_1 \ \mathcal{R}^* \ \mathbf{D}_2$ *and* $\mathbf{D}_2$ *is a covering of* $\mathbf{D}'_2$ *via* γ.

As a direct consequence of this lemma and of Proposition 1, if $\mathbf{D}_2$ is a proper covering of $\mathbf{D}'_2$, each label that appears in $\mathbf{D}'_2$ appears at least twice in $\mathbf{D}_2$ and therefore, we have the following result.

Proposition 3. *Let* $\mathbf{G}$ *be an undirected labelled graph. Let* Num *be a port numbering of* $\mathbf{G}$. *If the labelled digraph* $(\overleftrightarrow{\mathbf{G}}, \kappa, Num)$ *is not covering prime then there is no election algorithm and no naming algorithm for the graph* $\mathbf{G}$ *with* Num *as port numbering using* $Inst$ *as set of basic instructions.*

The election algorithm must work whatever the symmetric port numbering is. Let $(\overleftrightarrow{\mathbf{G}}, \kappa)$ be a covering of $(\overleftrightarrow{\mathbf{G}}', \kappa')$, and let Num be a local enumeration of arcs incident to vertices labelled **process** in the graph $(\overleftrightarrow{\mathbf{G}}', \kappa')$. The labelling Num induces a port numbering of $(\overleftrightarrow{\mathbf{G}}, \kappa)$ which is not necessarily symmetric (see the example in Figure 3).

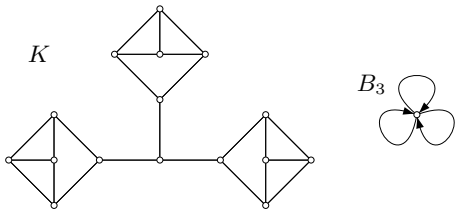

Fig. 3. There exists exactly one digraph B_3 such that $Dir(K)$ is a covering of B_3 : it is the 3-bouquet (the digraph with one node and three self-loops [BV02]). Thus $\mathcal{E}'(Dir(K))$ is a covering of $\mathcal{E}'(B_3)$ (Remark 1). It is easy to verify that no lifting of a local enumeration of arcs of $\mathcal{E}'(B_3)$ gives a symmetric port numbering of $\mathcal{E}'(Dir(K))$. Thus $Dir(K)$ is symmetric covering prime although it is not covering prime (see Definition 4). It follows from Theorem 1 that a naming algorithm exists for the graph K.

Before the next propositions we need two definitions [BV02]:

Definition 3. *Let* $\mathbf{D}_1$ *and* $\mathbf{D}_2$ *be two symmetric labelled digraphs, let* Sym_1 *and* Sym_2 *be symmetric relations of* $\mathbf{D}_1$ *and* $\mathbf{D}_2$, $\mathbf{D}_1$ *is a covering of* $\mathbf{D}_2$ *modulo* Sym_1 *and* Sym_2 *if there exists a morphism* φ *such that* $\mathbf{D}_1$ *is a covering of* $\mathbf{D}_2$ *via* φ *and* $\varphi \circ Sym_1 = Sym_2 \circ \varphi$.

Definition 4. *Let* $\mathbf{D}_1$ *be a symmetric digraph,* $\mathbf{D}_1$ *is symmetric covering prime if whenever there exists a symmetric relation* Sym_1 *of* $\mathbf{D}_1$, *a symmetric digraph* $\mathbf{D}_2$ *with a symmetric relation* Sym_2 *of* $\mathbf{D}_2$ *such that* $\mathbf{D}_1$ *is a covering of* $\mathbf{D}_2$ *modulo* Sym_1 *and* Sym_2 *then* $\mathbf{D}_1$ *is isomorphic to* $\mathbf{D}_2$.

From these definitions, there exists a symmetric port numbering Num of $\mathbf{G}$ such that $(\overleftrightarrow{\mathbf{G}}, \kappa, Num)$ is not covering prime if and only if $Dir(\mathbf{G})$ is not symmetric covering prime. Finally:

Proposition 4. *Let* $\mathbf{G}$ *be an undirected graph. If the labelled digraph* $Dir(\mathbf{G})$ *is not symmetric covering prime then there is no election algorithm and no naming algorithm for the graph* $\mathbf{G}$ *using* $Inst$ *as set of basic instructions.*

As immediate consequences of this result we deduce two classical results: there exists no deterministic election algorithm in an anonymous network of two processes that communicate by asynchronous message passing ([Tel00] p. 316) and more generally there exists no deterministic algorithm for election in an anonymous ring of known size ([Tel00] Theorem 9.5 p. 317) (sketches of the proofs are given in Figure 1 and in Figure 2).

5 A Mazurkiewicz-like Algorithm

The aim of this section is to prove the main result of this work:

Theorem 1. *Let* $\mathbf{G}$ *be a graph. There exist an election algorithm and a naming algorithm for* $\mathbf{G}$ *if and only if* $Dir(\mathbf{G})$ *is symmetric covering prime.*

The necessary part is Proposition 4, the following algorithm proves the other part. In [Maz97] Mazurkiewicz presents a distributed enumeration algorithm for non-ambiguous graphs (see also [GMM04]). The computation model in [Maz97] allows relabelling of all vertices in balls of radius 1. In the following we adapt Mazurkiewicz algorithm to graphs with port numbering and using $Inst$ as set of basic instructions. We shall denote our algorithm $\mathcal{M}$.

Description of $\mathcal{M}$. We first give a general description of the algorithm $\mathcal{M}$ applied to a labelled graph $\mathbf{G}$ equipped with a port numbering Num. We assume that $\mathbf{G}$ is connected. Let $\mathbf{G} = (G, \lambda)$ and consider a vertex v_0 of G, and the set $\{v_1, ..., v_d\}$ of neighbours of v_0. During the computation, each vertex v_0 will be labelled by a pair of the form $(\lambda(v_0), c(v_0))$, where $c(v_0)$ is a triple $(n(v_0), N(v_0), M(v_0))$ representing the following information obtained during the computation (formal definitions are given below): $n(v_0) \in \mathbb{N}$ is the *number* of the vertex v_0 computed by the algorithm, $N(v_0) \in \mathcal{N}$ is the *local view* of v_0, this view can be either empty or it is a set of the form: $\{((n(v_i), p_{s,i}, p_{r,i}), \lambda(v_i)) | 1 \leq i \leq d\}$, $M(v_0) \subseteq L \times \mathbb{N} \times \mathcal{N}$ is the *mailbox* of v_0 containing the whole information received by v_0 at previous computation steps. Let $(((n(v_i), p_{s,i}, p_{r,i}), \lambda(v_i)) 1 \leq i \leq d)$ be the local view of v_0. For each i, $(n(v_i), p_{s,i}, p_{r,i}))$ encodes a neighbour v_i of v_0, where: $n(v_i)$ is the number of v_i, v_i has sent its number to v_0 via the port $p_{s,i}$, and v_0 has received this message via the port $p_{r,i}$. Each vertex v gets information from its neighbours via messages and then attempts to calculate its own number $n(v)$, which will be an integer between 1 and $|V(G)|$. If a vertex v discovers the existence of another vertex u with the same number, then it compares its own label and its own local view with the label and the local view of u. If the label of

u or the local view of u is "stronger", then v chooses another number. Each new number, with its local view, is broadcasted again over the network. At the end of the computation, it is not guaranteed that every vertex has a unique number, unless the graph $(\overleftrightarrow{\mathbf{G}}, \kappa, Num)$ is covering prime. However, all vertices with the same number will have the same label and the same local view.

Algorithm 1: The algorithm $\mathcal{M}$.

Var : $n(v_0)$: integer **init** 0 ;
$N(v_0)$: set of local view **init** $\emptyset$;
N : set of local view ;
$M(v_0)$: mailbox **init** $\emptyset$;
M, M_a : mailbox;
$\lambda(v_0), c_a, l$: element of L;
i, x, p, q, n_a : integer;

$\mathbf{I}_0$: $\{n(v_0) = 0$ and no message has arrived at $v_0\}$
begin
  $n(v_0) := 1$;
  $M(v_0) := \{(\lambda(v_0), 1, \emptyset)\}$;
  for $i := 1$ **to** $deg(v_0)$ **do** send$< (n(v_0), M(v_0)), i >$ via port i ;
end

$\mathbf{R}_0$: {A message $<$ **mes**$= (n_a, M_a)$, $p >$ has arrived at v_0 from port $q\}$
begin
  $M := M(v_0)$;
  $M(v_0) := M(v_0) \cup M_a$;
  if $((x, p, q) \notin N(v_0)$ *for some* $x)$ **then**
    $N(v_0) := N(v_0) \cup \{(n_a, p, q)\}$;
  if $((x, p, q) \in N(v_0)$ *for some* $x < n_a)$ **then**
    $N(v_0) := (N(v_0) \setminus \{(x, p, q)\}) \cup \{(n_a, p, q)\}$;
  if $(n(v_0) = 0)$ *or* $(n(v_0) > 0$ *and there exists* $(l, n(v_0), N) \in M(v_0)$
  such that $(\lambda(v_0) <_L l)$ *or* $((\lambda(v_0) = l)$ *and* $(N(v_0) \prec N)))$ **then**
    $n(v_0) := 1 + \max\{n \in \mathbb{N} \mid (l, n, N) \in M(v_0)$ for some $l, N\}$;
    $M(v_0) := M(v_0) \cup \{(\lambda(v_0), n(v_0), N(v_0))\}$;
  if $(M(v_0) \neq M))$ **then**
    for $(i := 1$ **to** $deg(v_0))$ **do** send $< (n(v_0), M(v_0)), i >$ via port i;
end

An Order on Local Views. We assume for the rest of this paper that the set of labels L is totally ordered by $<_L$. Consider a vertex v such that the local view $N(v) \in \mathcal{N}$ is the set $\{(n_1, p_{s,1}, p_{r,1}), (n_2, p_{s,2}, p_{r,2}), \ldots, (n_d, p_{s,d}, p_{r,d})\}$. We assume that for each $i < d$, $(n_{i+1}, p_{s,i+1}, p_{r,i+1}) <_{Lex} (n_i, p_{s,i}, p_{r,i})$ where $<_{Lex}$ denotes the usual lexical order. We say that $((n_1, p_{s,1}, p_{r,1}), (n_2, p_{s,2}, p_{r,2}), \ldots,$ $\ldots (n_d, p_{s,d}, p_{r,d}))$ is the ordered representation $N_>(v_0)$ of the local view of v_0. Let $\mathcal{N}_>$ be the set of such ordered tuples. We define a total order $\prec$ on $\mathcal{N}_>$ using the alphabetical order that induces naturally a total order on $\mathcal{N}$. This order can also be defined on $\mathcal{N}$ as follows: $N_1 \prec N_2$ if the maximal element for the lexical

order $<_{Lex}$ of the symmetric difference $N_1 \triangle N_2 = N_1 \cup N_2 \setminus N_1 \cap N_2$ belongs to N_2. If $N(u) \prec N(v)$, then we say that the local view $N(v)$ of v is stronger than the one of u.

The Final Labelling. Let $\mathbf{G} = (G, \lambda)$ be a connected labelled graph with the port numbering Num. If v is a vertex of G then the label of v after a run ρ of $\mathcal{M}$ is denoted $(\lambda(v), c_\rho(v))$ with $c_\rho(v) = (n_\rho(v), N_\rho(v), M_\rho(v))$ and (λ, c_ρ) denotes the final labelling. Finally $\mathcal{M}$ verifies:

Proposition 5. *Any run ρ of $\mathcal{M}$ on $\mathbf{G} = (G, \lambda)$, a connected labelled graph with the port numbering Num, terminates and yields a final labelling (λ, c_ρ) verifying the following conditions for all vertices v, v' of G:*

1. *there exists an integer $k \leq V(G)$ such that $\{n_\rho(v) \mid v \in V(G)\} = [1, k]$.*
2. *$M_\rho(v) = M_\rho(v')$.*
3. *$(\lambda(v), n_\rho(v), N_\rho(v)) \in M_\rho(v')$.*
4. *Let $(l, n, N) \in M_\rho(v')$. Then $\lambda(v) = l$, $n_\rho(v) = n$ and $N_\rho(v) = N$ for some vertex v if and only if there is no triple $(l', n, N') \in M_\rho(v')$ with $l <_L l'$ or $(l = l'$ and $N \prec N')$.*
5. *$n_\rho(v) = n_\rho(v')$ implies $(\lambda(v) = \lambda(v')$ and $N(v) = N(v'))$.*

Consider a graph $\mathbf{G}$ that is symmetric covering prime. For each port numbering Num, the graph $(\overleftrightarrow{\mathbf{G}}, \kappa, Num)$ is covering prime and then from Proposition 5, at the end of the computation, each vertex $v \in V(G)$ has a unique number $n(v)$. Moreover, once a vertex gets a number $n(v) = |V(G)|$, it knows that all the vertices have a unique identifier, it can take the label *elected* and broadcast the information. Theorem 1 follows from Proposition 5 and the impossibility results of the previous section.

Remark 3. The proof of this proposition uses increasing properties and invariant properties as in [Maz97]. In particular, the number $n(v)$ (resp. the mailbox $M(v)$) can only increase for the order $\leq$ (resp. for $\subseteq$) during the computation. Consequently, if a message $m_1 = (n_1(v), M_1(v), p)$ has been sent before $m_2 = (n_2(v), M_2(v), p)$ by a vertex v to a node w is such that m_2 arrives before m_1, then when the message m_1 is read by w, $M_1(v) \subsetneq M_2(v) \subseteq M(w)$ and $n_1(v) \leq n_2(v)$. Consequently, this message does not modify the state of the vertex w and can be considered as ignored by the vertex w. We can therefore deduce that Algorithm 1 does not require ordering of messages, that is, it does not require that messages are received in the same order that they have been sent.

Remark 4. Note that not all the elements of $M(v)$ are useful during the whole computation. In fact, for all $(n, l_1, N_1), (n, l_2, N_2) \in M(v)$, if $l_1 <_L l_2$ or $l_1 = l_2$ and $N_1 \prec N_2$, we can remove (n, l_1, N_1) from $M(v)$. Consequently, if we remove all such elements of $M(v)$, we get for each number n exactly one element in $M(v)$. If we can encode the labels l of L with $O(\log |V(G)|)$ bits, then the size of the mailbox of v is $O(|E(G)| \log |V(G)|)$ and therefore, the size of the messages is also $O(|E(G)| \log |V(G)|)$. Moreover, we can show that the total number of messages sent during the computation is $O(|E(G)|^2 |V(G)|)$ and therefore the amount of information sent all over the network during the computation is polynomial in the size of the network.

References

[Ang80] D. Angluin. Local and global properties in networks of processors. In *Proceedings of the 12th Symposium on Theory of Computing*, pages 82–93, 1980.

[AW98] H. Attiya and J. Welch. *Distributed computing: fundamentals, simulations, and advanced topics*. McGraw-Hill, 1998.

[BCG+96] P. Boldi, B. Codenotti, P. Gemmell, S. Shammah, J. Simon, and S. Vigna. Symmetry breaking in anonymous networks: Characterizations. In *Proc. 4th Israeli Symposium on Theory of Computing and Systems*, pages 16–26. IEEE Press, 1996.

[BV99] P. Boldi and S. Vigna. Computing anonymously with arbitrary knowledge. In *Proceedings of the 18th ACM Symposium on principles of distributed computing*, pages 181–188. ACM Press, 1999.

[BV01] Paolo Boldi and Sebastiano Vigna. An effective characterization of computability in anonymous networks. In Jennifer L. Welch, editor, *Distributed Computing. 15th International Conference, DISC 2001*, volume 2180 of *Lecture Notes in Computer Science*, pages 33–47. Springer-Verlag, 2001.

[BV02] P. Boldi and S. Vigna. Fibrations of graphs. *Discrete Math.*, 243:21–66, 2002.

[Cha05] J. Chalopin. Election and local computations on closed unlabelled edges (*extended abstract*). In *Proc.of SOFSEM 2005*, number 3381 in LNCS, pages 81–90, 2005.

[CM04] J. Chalopin and Y. Métivier. Election and local computations on edges (*extended abstract*). In *Proc. of Foundations of Software Science and Computation Structures, FOSSACS'04*, number 2987 in LNCS, pages 90–104, 2004.

[CMZ04] J. Chalopin, Y. Métivier, and W. Zielonka. Election, naming and cellular edge local computations (*extended abstract*). In *Proc. of International conference on graph transformation, ICGT'04*, number 3256 in LNCS, pages 242–256, 2004.

[FR03] F. Fich and E. Ruppert. Hundreds of impossibility results for distributed computing. *Distributed computing*, 16:121–163, 2003.

[GMM04] E. Godard, Y. Métivier, and A. Muscholl. Characterization of Classes of Graphs Recognizable by Local Computations. *Theory of Computing Systems*, (37):249–293, 2004.

[LeL77] G. LeLann. Distributed systems: Towards a formal approach. In B. Gilchrist, editor, *Information processing'77*, pages 155–160. North-Holland, 1977.

[Maz97] A. Mazurkiewicz. Distributed enumeration. *Inf. Processing Letters*, 61:233–239, 1997.

[RFH72] P. Rosenstiehl, J.-R. Fiksel, and A. Holliger. Intelligent graphs. In R. Read, editor, *Graph theory and computing*, pages 219–265. Academic Press (New York), 1972.

[Tel00] G. Tel. *Introduction to distributed algorithms*. Cambridge University Press, 2000.

[YK96] M. Yamashita and T. Kameda. Computing on anonymous networks: Part i - characterizing the solvable cases. *IEEE Transactions on parallel and distributed systems*, 7(1):69–89, 1996.